class
SOFTWARE
ACADÉMICO

Prueba técnica programador React Js

Innovasoft S.A

Contenido

Objetivo

Diseñar un pequeño aplicativo que permita dar solución y representación gráfica a un problema clásico de **Mantenimiento de un Clientes**.

Lineamientos técnicos

General

- Recordar realizar todas las validaciones pertinentes de manejo de tipos de datos y extensiones de estos.

Ide

- **Visual Studio CODE** : Por ligereza y dada la naturaleza de la prueba este es el Editor de código para la prueba.

Frontend

- Debe ser desarrollado en **React Js v17**
  - Los componentes deben ser desarrollados en **ECMAScript 6** como base del JS
  - Creación de Componentes Basados en **Componentes Funcionales y React Hooks (useEffect , useState,useReducer,useContext)**
  - El manejo del estado deben realizarlo usando ContextApi, que es el manejador de estado que viene en las últimas versiones de React JS
  - Para el consumo de servicios **Api Rest** debe realizarse por medio de **Axios** haciendo mucho uso de programación **asíncrona** con **Promesas** o **async y await**
  - La navegación entre los componentes debe realizarse por medio de **React Router Dom**

- El tipo de proyecto frontend debe ser un **SPA**
- El proyecto de frontend debe ser creado con la plantilla **npx create-reactapp** de **NPM package Manager**
- Todas las dependencias del frontend debe ser instaladas y manejadas con **NPM package Manager**

Marco de diseño

- Todo el diseño debe ser sustentado en los componentes y normativas de Material UI (https://material-ui.com/es/) como implementación concreta de material UI
- Debe de tomar en cuenta los principios generales establecidos en Material Desing de Google (https://material.io/design)
- Debe tomar en cuenta que el prototipado del diseño debe ser responsive
- La temática del esquema de colores debe evidenciar un estilo ejecutivo

Contexto de la necesidad

General

La empresa tiene la necesitad de plantear un mantenimiento para los clientes que maneja la institución que permita

1. Registrar
2. Listar
3. Detalle
4. Actualizar
5. Consultar
6. Eliminar

Detallado

- Todo el proceso gira en torno a los de un **Cliente** que tiene los siguientes campos:
    - Nombre
    - Apellidos
    - Identificación
    - Teléfono Celular
    - Teléfono Otros
    - Dirección
    - Fecha de Nacimiento
    - Fecha de Afiliación
    - Sexo (Masculino-Femenino)
    - Reseña Personal
    - Imagen (Imagen del Cliente) (Opcional)
    - Intereses (Lista desplegable)

- Se deberá diseñar y programar el Detalle que se presenta en la sección del prototipado, un pequeño sitio que tenga su sección de Frontend en React Js.

Detalle por Componente

- **Login**
  - Página para inicio de sesión
- **Registro**
  - Página para registro de usuarios del sistema
- **Página de error**
  - Presentar la página de error que se presentaría en caso de que se desee acceder a una página que **no** exista en el sistema
- **Home**
  - Página de entrada al sitio cuando se tenga un login exitoso
- **Consulta Clientes**
  - Debe contener un listado en el cual se presente la información de los clientes de la empresa
  - Deberá presentar una sección de filtros para la información del listado
  - Debe proporcionar un mecanismo de salida para poder ver la información de un cliente determinado y poder editar su información
  - Debe tener el mecanismo de interacción más apropiado para permitir realizar las operaciones básicas que se ejecutan sobre el usuario
    - Crear
    - Eliminar
    - Detalle

- **Mantenimiento Clientes**
  - Se encarga de realizar el proceso de registro de nuevos clientes que va a tener la institución o bien la actualización de un cliente existente

Detalles de los componentes

Los detalles proporcionados son de carácter ilustrativo a fin de que el desarrollador tenga una idea de los esperado en el proceso de diseño y maquetado en React JS y de comportamiento para el consume del api para gestionar la información en la base de datos.

Para las peticiones al API debe utilizar la URL base:

https://pruebareactjs.test-class.com/Api/

Ver endpoints:

https://pruebareactjs.test-class.com/Api/swagger/index.html

Detalle Páginas

Login

Comportamiento:

- Es obligatorio el ingreso del usuario y contraseña, en caso de que no ingrese alguno de estos datos se debe mostrar un mensaje al usuario que los datos son requeridos.
- “Recuérdame” debe permitir guardar el nombre del usuario para que al volver a cargar la página de login no sea necesario volver a digitar este dato.

- Si no tiene un usuario deberá crear una cuenta por medio del link de “¿No tiene una cuenta? Regístrese”.
- Al presionar “INICIAR SESIÓN” se debe llamar al endpoint, el cual si los credenciales son validados le retornará un json con los siguientes datos, los cuales necesitará más adelante:

URL Endpoint **(Post)**: **URL base** + “**api/Authenticate/login**”
Ejemplo: https://pruebareactjs.test-class.com/Api/api/Authenticate/login

Petición:

```
{
  "username": "string",
  "password": "string"
}
```

Respuesta 200 Success:

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9
yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1lIjoiYXJpdmVsIiwianRpIjoiYjA2
ZDYxNzItZGQ4MC00ZTkxLTljYzYtODU5NmI0Y2QwMWY5IiwiaHR0cDovL3NjaGVtYXMubWljc
m9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoiQWRtaW4iLCJleH
AiOjE2NTExMTcxNzIsImlzcyI6Imh0dHA6Ly9sb2NhbGhvc3Q6NjE5NTUiLCJhdWQiOiJodHR
wOi8vbG9jYWxob3N0OjQyMDAifQ.LL_k8-oG4EsF8ba3nocf5tpK4WfXaXOOUKcEVm3Kdo8",
  "expiration": "2022-04-28T03:39:32Z",
  "userid": "c8c8b7f9-edd5-4b0b-b488-74d82747ac56",
  "username": "arivel"
}
```

El token y userid es necesario para las peticiones a los demás endpoints del Api.

Registro

Comportamiento:

- Todos los campos son obligatorios.
- Se debe validar que ingrese un correo válido.
- La contraseña debe ser mayor a 8 y menor o igual a 20 caracteres, debe tener números, al menos una mayúscula y una minúscula.
- Al presionar “REGISTRARSE” se debe llamar al endpoint, el cual si los datos son validados le retornará un json con los siguientes datos:

URL Endpoint **(Post)**: **URL base** + “**api/Authenticate/register**”

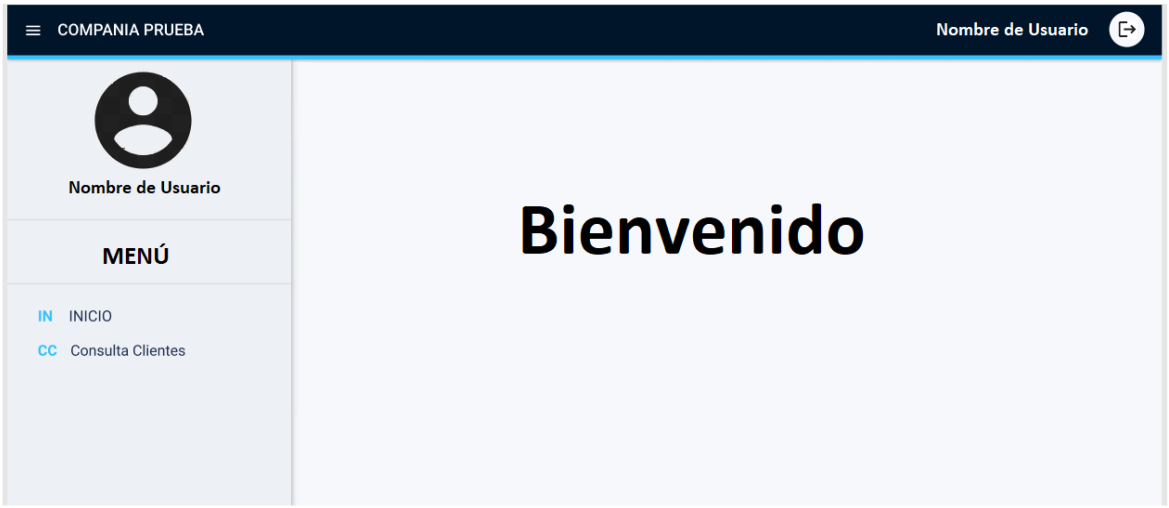
Petición:

```
{
  "username": "string",
  "email": "user@example.com",
  "password": "string"
}
```

Respuesta 200 Success:

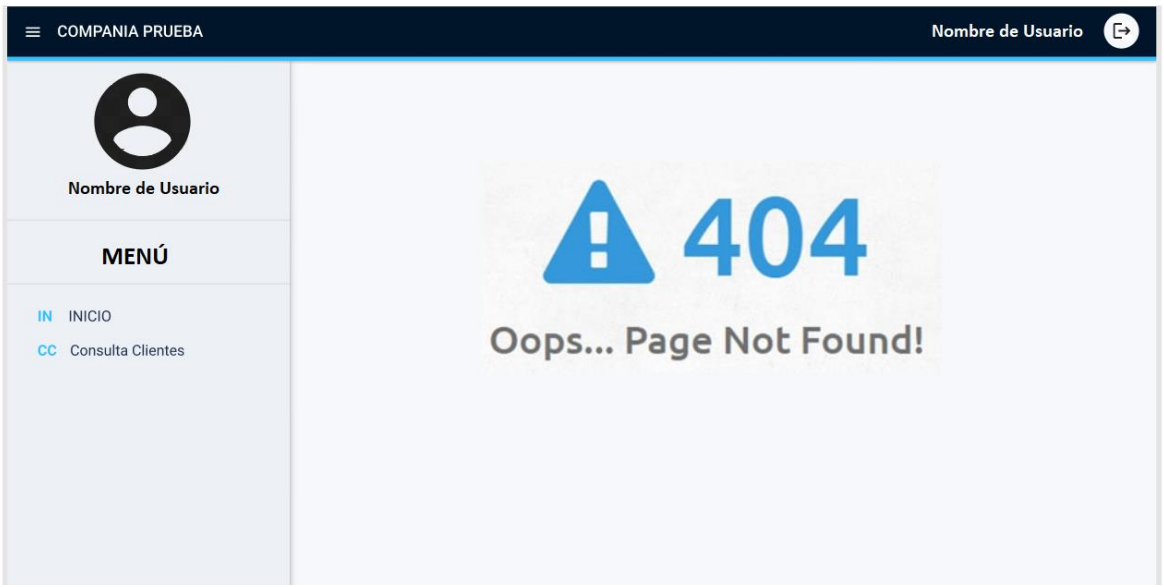
```
{
  "status": "Success",
  "message": "Usuario creado correctamente"
}
```

Home

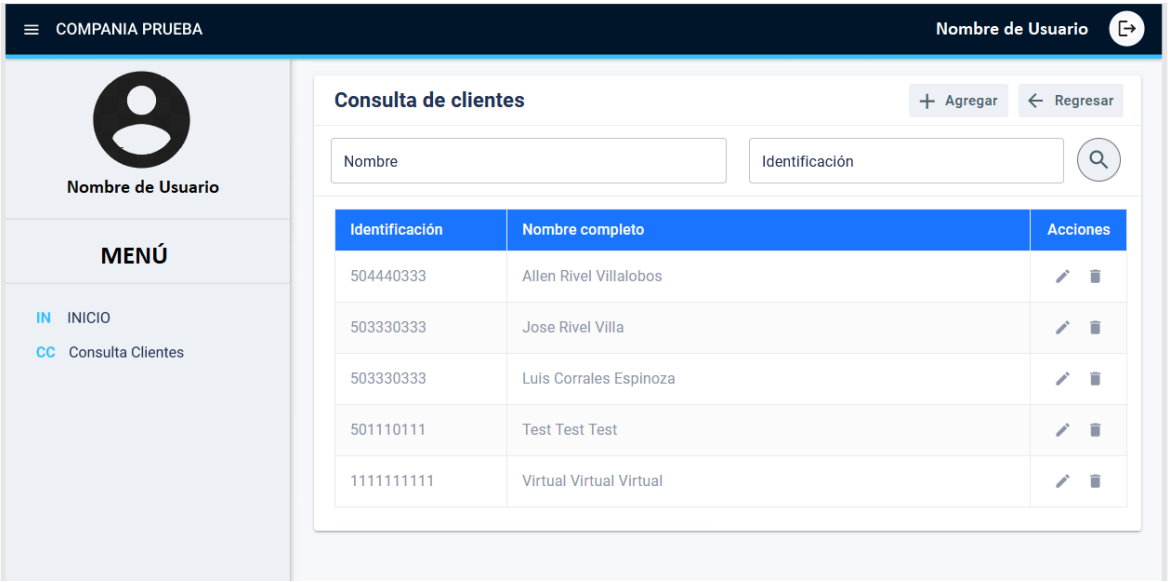


Es la pantalla al ingresar del login, debe tener la opción de cuentas clientes para redireccionar a la consulta de clientes, también el menú INICIO debe regresar a esta página de home si se está mostrando otro componente, adicional debe mostrar el nombre del usuario en sesión tanto en la barra superior como en el menú lateral, debe permitir cerrar sesión (botón a la derecha de la barra superior)

Página de error



Consulta Clientes



Comportamiento:

- Para buscar clientes existentes utiliza el botón buscar (Lupa).
- Para agregar un nuevo cliente debe utilizar el botón de Agregar (redirecciona al componente de Mantenimiento Cliente).
- Para editar un cliente determinado debe utilizar el botón de editar (Lápiz) (redirecciona al componente de Mantenimiento Cliente y carga la información del cliente).
- El icono de eliminar debe borrar el cliente seleccionado.
- El botón de Regresar debe volver al home.
- Al eliminar se debe mostrar un modal o popup que le solicite al usuario la confirmación para eliminar el registro.

URL Endpoint **Buscar (Post)**: **URL base** + "**api/Cliente/Listado**" (autorización por Bearer Token)
Petición:

```
{
  "identificacion": "string", (no es obligatorio)
  "nombre": "string", (no es obligatorio)
  "usuarioId": "string" (id del usuario en sesión)
}
```

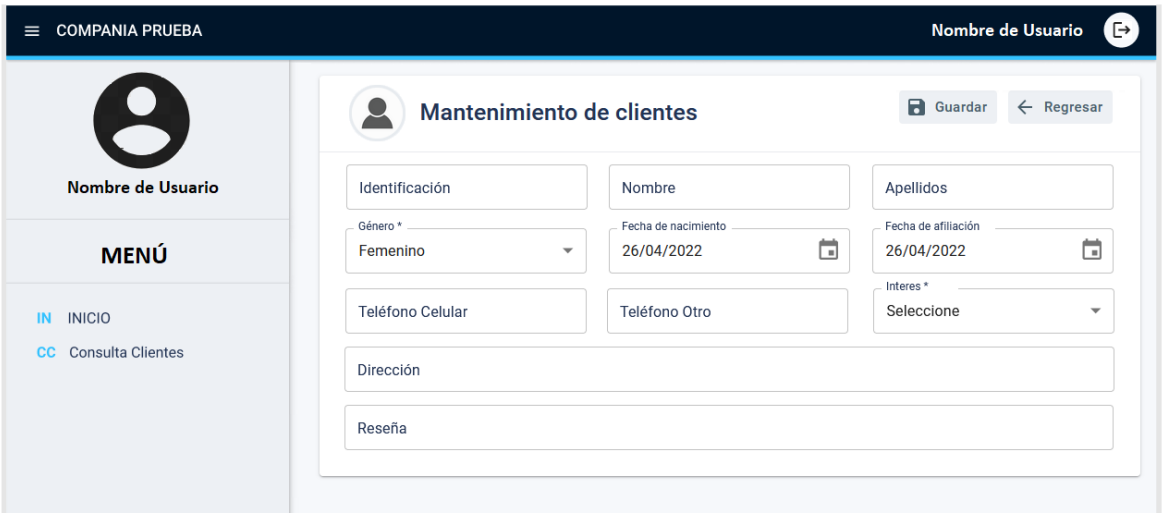
Retorna (arreglo):

```
[
  {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "identificacion": "string",
    "nombre": "string",
    "apellidos": "string"
  }
]
```

URL Endpoint **Eliminar (Delete)**: **URL base** + "**api/Cliente/Eliminar/{IdCliente}**" (autorización por Bearer Token)

Respuesta: 200 Success

Mantenimiento Clientes



Comportamiento:

- Todos los campos son obligatorios (deben mostrar * en la etiqueta) excepto la imagen del cliente.
- Para mostrar y cargar imágenes se debe utilizar el componente de imagen que está a la par del título “Mantenimiento de clientes”.
- Primero se debe cargar el listado de Intereses.
- Si es para editar un cliente existente se debe cargar la información de este.
- El botón Guardar permite crear o actualizar un cliente según sea el caso.
- El botón de regresar debe volver a la página de consulta clientes.
- Los datos de la imagen deben manejarse en base64, esta no es obligatoria.
- Para el Género se maneja M para Masculino y F para Femenino.
- Luego de crear o actualizar un cliente existente se debe redirigir a la página de consulta de clientes.

URL Endpoint **Carga Listado (Get): URL base** + “**api/Intereses/Listado**”

Petición: no recibe parámetros
Respuesta (arreglo):

```
[
  {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "descripcion": "string"
  }
]
```

URL Endpoint **Obtener Info Cliente (Get): URL base** + “**/api/Cliente/Obtener/{IdCliente}**”
Respuesta 200 Success:

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

```
  "nombre": "string", (tamaño máximo 50)
  "apellidos": "string", (tamaño máximo 100)
  "identificacion": "string", (tamaño máximo 20)
  "telefonoCelular": "string",(tamaño máximo 20)
  "otroTelefono": "string",(tamaño máximo 20)
  "direccion": "string",(tamaño máximo 200)
  "fNacimiento": "2022-04-27T05:27:43.365Z",
  "fAfiliacion": "2022-04-27T05:27:43.365Z",
  "sexo": "F/M",(tamaño máximo 1)
  "resenaPersonal": "string", (tamaño máximo 200)
  "imagen": "string", (base64 de la imagen)
  "interesesId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

URL Endpoint **Crear Cliente (Post): URL base** + “**/api/Cliente/Crear**”
Petición:

```
{
  "nombre": "string", (tamaño máximo 50)
  "apellidos": "string", (tamaño máximo 100)
  "identificacion": "string", (tamaño máximo 20)
  "telefonoCelular": "string",(tamaño máximo 20)
  "otroTelefono": "string",(tamaño máximo 20)
  "direccion": "string",(tamaño máximo 200)
  "fNacimiento": "2022-04-27",
  "fAfiliacion": "2022-04-27",
  "sexo": "F/M",(tamaño máximo 1)
  "resenaPersonal": "string", (tamaño máximo 200)
  "imagen": "string", (base64 de la imagen)
  "interesFK": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "usuarioId": "string" (Id usuario en sesión)
}
```

Respuesta: 200 Success

URL Endpoint **Actualizar Cliente (Post): URL base** + “**/api/Cliente/Actualizar**”
Petición:

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "nombre": "string",
  "apellidos": "string",
  "identificacion": "string",
  "celular": "string",
  "otroTelefono": "string",
  "direccion": "string",
  "fNacimiento": "2022-04-27T05:29:49.210Z",
  "fAfiliacion": "2022-04-27T05:29:49.210Z",
  "sexo": "string",
  "resennaPersonal": "string",
  "imagen": "string",
  "interesFK": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
  "usuarioId": "string" (Id usuario en sesión)
}
```

Respuesta: 200 Success

NOTAS IMPORTANTES

1. Si se presenta algún problema cuando se realiza una petición al API se debe mostrar un mensaje al usuario que hubo un inconveniente con la transacción.
2. Luego de crear/registrar, actualizar o eliminar se debe mostrar un mensaje al usuario indicando que el proceso se realizó correctamente.
3. Para mostrar los mensajes se puede utilizar los Snackbar de Material UI.